Keyframe-centric Trajectory Diffusion: Learning Latent Representations for Conditional Robotic Motion Planning

*Note: Sub-titles are not captured for https://ieeexplore.ieee.org and should not be used

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—Generating smooth, collision-free trajectories for robots in cluttered 3D environments is a critical bottleneck for achieving autonomy. Mainstream learning-based paradigms, despite their promise, fall into two categories with distinct limitations. The first, generative modeling (e.g. diffusion models), operates directly in the high-dimensional redundant joint space. This approach is not only computationally expensive, but also fundamentally structurally unaware, struggling to capture the intrinsic kinematic and temporal structure of trajectories. The second, direct policy learning (e.g., behavior cloning), while reactive, typically learns deterministic policies, which limits solution diversity and hinders generalization. This paper proposes a fundamental shift: conditional trajectory generation in a learned, low-dimensional semantic latent space. To this end, we design an innovative Keyframe Transformer-VAE (KT-VAE) architecture that adaptively compresses long-horizon trajectories into a compact set of semantic keyframes via an attention mechanism. Building upon this representation, we train a conditional diffusion model that takes raw obstacle point clouds and task endpoints as conditions, achieving end-to-end generation of keyframe sequences. Experimental results demonstrate that our method significantly improves the planning success rate and trajectory quality, exhibiting excellent generalization to complex zero-shot scenarios. We also release a large-scale, procedurally generated, and physically consistent trajectory dataset to facilitate future research.**

***Index Terms*—Motion Planning, Diffusion Models, Representation Learning, Robotic Manipulation, Latent Space.**

I. Introduction

Autonomous robotic manipulation holds the key to transformative applications, from automating complex logistics in warehouses to assisting in domestic environments. Central to this autonomy is the challenge of motion planning: Generating smooth, feasible, and collision-free trajectories for high-degree-of-freedom (DOF) manipulators. This problem is defined by a high-dimensional configuration space fraught with complex constraints, including joint limits, self-collision avoidance, and interaction with a cluttered, often unstructured, 3D environment. Consequently, motion planning remains a significant bottleneck to deploying robots in the real world.

The classical paradigm for this problem typically involves a synergistic, two-stage pipeline. First, a sampling-based planner, such as RRT* or BIT*, provides global exploration to find a topologically valid, collision-free path. This initial path is then used to warm-start an optimization-based method like CHOMP, which refines it into a smooth, locally optimal trajectory. While this hybrid approach is robust, it suffers from several fundamental limitations. It is a computationally intensive and sequential process, ill-suited for real-time applications. Furthermore, the pipeline critically relies on access to a perfect world model—a precise, geometric representation of the environment for collision checking. Bridging the gap from raw sensor data to such a model is a significant challenge in itself. While diverse solutions can be obtained by repeatedly running the stochastic planner, this brute-force approach is computationally infeasible for practical use, failing to provide a set of high-quality options efficiently. Most importantly, this paradigm is a non-learning approach, solving every new problem from scratch without accumulating knowledge from past experience.

Generative models, particularly Denoising Diffusion Probabilistic Models (DDPMs), have recently emerged as a powerful paradigm. Initial works like EDMP demonstrated the viability

Identify applicable funding agency here. If none, delete this.

of diffusion for trajectory generation. Recognizing the inefficiency of operating in the full waypoint space, a significant conceptual leap was made by Motion Planning Diffusion (MPD), which pioneered the idea of performing diffusion in a lower-dimensional parametric space, specifically the control points of B-spline curves. This approach improves efficiency by reducing the dimensionality of the space being modeled. Concurrently, RobotDiffuse [from Zhang et al.] explored end-to-end generation by integrating a point cloud encoder with a Transformer-based denoising network.

Despite these advances, a crucial challenge remains: The representation used for diffusion is either a handcrafted mathematical parameterization (like B-spline coefficients) or the raw, redundant joint space itself. While B-splines reduce dimensionality, their control points lack clear semantic meaning regarding the trajectory's overall structure, and are not learned from data to be optimal for generation. Consequently, this fixed parameterization can impose a complex and structurally suboptimal landscape for the generative model to navigate.

We posit that the next frontier lies in moving from a parameterized space to a truly learned semantic latent space. In this paper, we introduce a novel framework centered on this idea. Our core contribution is a Keyframe Transformer-VAE (KT-VAE), an architecture designed not merely to compress trajectories, but to learn an optimal, low-dimensional representation corresponding to the trajectory's semantic keyframes. The choice of a Variational Autoencoder (VAE) architecture is foundational to our approach; its KL-divergence regularization term encourages the learned latent space to be continuous and well-structured, which in turn creates a suitable manifold for the diffusion model to operate on. The KT-VAE encoder innovatively employs a set of learnable query vectors with an attention pooling mechanism to autonomously discover and extract these keyframes from a full-length trajectory. This results in a compact and structured latent space where each element represents a high-level abstraction of a key motion primitive within the trajectory.

Within this superior latent space, we train a conditional diffusion model. Our system achieves end-to-end generation from raw perception to a high-level plan by conditioning on the output of a simple yet effective MLP-based point cloud encoder and the task specifications. To ensure physical validity when decoding from the latent space, we introduce a sophisticated hierarchical, physics-informed loss architecture and an intelligent Min-SNR-γ curriculum learning strategy. This framework effectively back-propagates physical constraints from the joint space to guide the generative process in the latent space.

Our main contributions can be summarized as follows.

1) We propose a novel Keyframe Transformer-VAE (KT-VAE) architecture that learns to compress trajectories into a low-dimensional sequence of semantic keyframes, providing a more effective representation for generative modeling than prior handcrafted parameterizations.
2) We present an end-to-end conditional latent diffusion framework that generates these keyframe sequences directly from raw point cloud observations and task goals, exhibiting strong generalization capabilities.
3) We design a hierarchical physics-informed training strategy, complemented by a Min-SNR-γ weighting scheme, that successfully enforces physical constraints (e.g., collision avoidance, smoothness) while operating within the latent space.
4) We introduce and will release a new, large-scale, procedurally generated dataset of physically-consistent trajectories in complex, cluttered environments, serving as a valuable resource for future research in learning-based motion planning.

II. RELATED WORK

The challenge of robotic motion planning has been a central theme in robotics research for decades, leading to a rich body of literature. Our work builds upon insights from classical planning, learning-based methods, and generative modeling. In this section, we situate our contributions within these domains.

A. Classical Motion Planning

Classical motion planning approaches can be broadly categorized into sampling-based and optimization-based methods. Sampling-based planners, such as Probabilistic Roadmaps (PRM) [**?**] and Rapidly-exploring Random Trees (RRT) [**?**], have been highly influential due to their ability to handle high-dimensional configuration spaces. They offer probabilistic completeness, with variants like RRT* [**?**] and BIT* [**?**] further providing asymptotic optimality guarantees. However, their primary drawback is the computational cost associated with random sampling, which often scales poorly in complex or narrow-passage environments, making them ill-suited for real-time applications.

Optimization-based methods, such as CHOMP [**?**] and TrajOpt [**?**], formulate motion planning as a trajectory optimization problem. They can produce smooth, high-quality paths but are sensitive to initialization and are prone to converging to poor local minima, especially in non-convex settings with complex obstacles. The limitations of these classical methods, particularly in terms of computational efficiency and robustness, have motivated the increasing adoption of data-driven, learning-based techniques.

B. Learning-based Motion Planning

Learning-based approaches leverage data from expert demonstrations or experience to accelerate the planning process and improve generalization.

1) Direct Policy Learning: A prominent line of work involves learning a direct mapping from perception to action. Motion Policy Networks (MπNets) [**?**] pioneered an end-to-end approach, learning a reactive policy that maps raw point cloud observations to robot motions. This enables extremely fast inference, which is advantageous for dynamic environments. However, such methods primarily learn a deterministic or unimodal policy. For a given planning problem, they tend to produce a single solution, lacking the ability to generate

a diverse set of valid trajectories. Furthermore, as reactive, local policies, they can struggle in environments with deep local minima (e.g., U-shaped obstacles) where a global understanding of the scene is required. In contrast, our work focuses on a generative paradigm, capable of producing a rich distribution of globally coherent and feasible trajectories for a single planning query.

2) Generative Models for Motion Planning: The most relevant recent works have explored deep generative models, particularly Denoising Diffusion Probabilistic Models (DDPMs), for trajectory generation. Methods like Motion Planning Diffusion (MPD) [**?**] and Ensemble-of-costs-guided Diffusion for Motion Planning (EDMP) [**?**] have successfully demonstrated that diffusion models can generate diverse and smooth robot trajectories. These models are typically conditioned on start and goal configurations and can be guided by differentiable cost functions, such as collision penalties, to satisfy constraints.

However, a fundamental limitation of these pioneering works is that they operate **directly in the high-dimensional and often redundant joint space** (e.g., $\mathbb{R}^{50\times7}$). This approach presents several significant challenges:

- Computational Inefficiency: The diffusion and denoising process is applied to a high-dimensional vector at every timestep, leading to substantial computational overhead.
- Structural Corruption: The forward diffusion process adds noise indiscriminately to every joint at every timestep, corrupting the inherent kinematic and temporal structure of the trajectory. The denoising network must then expend a significant portion of its capacity to re-learn these fundamental physical correlations.
- Complexity of Control: Guiding or conditioning a high-dimensional sequence is inherently complex, making it difficult to enforce global consistency or make targeted modifications.

Our work addresses these limitations head-on by proposing a fundamental paradigm shift. Instead of diffusing the raw trajectory, we first learn a compact, structured latent representation of **semantic keyframes** and perform the conditional diffusion process within this low-dimensional and information-rich space. This key difference allows for more efficient learning, better preservation of trajectory structure, and a more potent mechanism for conditioning.

C. Representation Learning for Sequential Data

The concept of learning a compressed representation is central to our approach. Variational Autoencoders (VAEs) [**?**] have been widely used to learn meaningful latent spaces for complex data. In parallel, the Transformer architecture [**?**] has revolutionized sequence modeling by effectively capturing long-range dependencies through its self-attention mechanism. These models have found immense success in domains like natural language processing, computer vision, and speech synthesis.

While Transformers and VAEs are established tools, their application to robotic trajectory representation is an active area of research. Our work introduces a novel synthesis of these principles. The proposed Keyframe Transformer-VAE (KT-VAE) is not a standard application; its architecture, featuring **learnable queries for attention pooling** in the encoder and a hybrid **convolutional-transformer decoder**, is specifically designed to discover and model the unique spatio-temporal structure of robotic motion. It learns not just a compressed code, but a representation where each element in the latent sequence corresponds to a semantically meaningful "keyframe" of the full trajectory.

D. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

E. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there

won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

F. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [7].

G. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

H. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

I. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

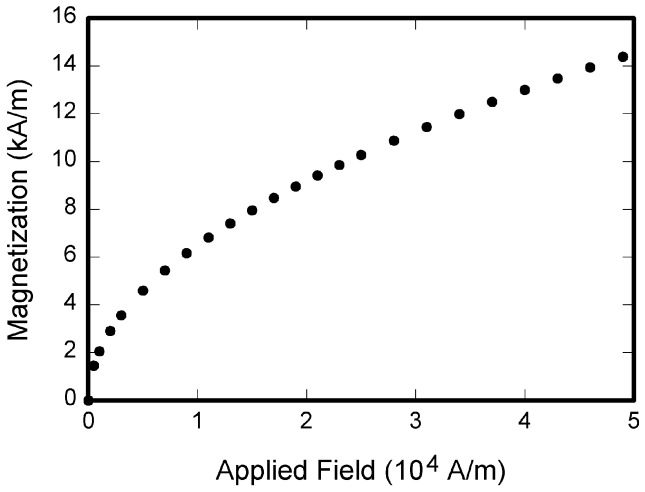

Fig. 1. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization

{A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

III. METHODOLOGY

This section details our framework for robotic motion planning, which learns to generate trajectories in a semantic latent space. Our approach is founded on the principle that by learning a superior, low-dimensional representation of trajectories, the subsequent task of conditional generative modeling becomes significantly more tractable and effective. We begin by reformulating the classical motion planning problem as a probabilistic inference task. We then introduce our core architectural innovation, the Keyframe Transformer-VAE (KT-VAE), designed to learn this representation. Finally, we describe the conditional diffusion model that operates within this latent space and the hierarchical training strategy that ensures the physical validity of the generated motions.

A. From Trajectory Optimization to Generative Inference

Traditionally, robotic motion planning is formulated as a trajectory optimization problem. Given a planning context $O = (q_{\text{start}}, q_{\text{goal}}, P_{\text{obs}})$, the goal is to find an optimal trajectory $\tau^* = \{q_t\}_{t=1}^T$ that minimizes a cost function $C(\tau)$:

$$\tau^* = \arg\min_\tau C(\tau). \tag{2}$$

The cost function is typically a weighted sum of terms penalizing undesirable properties, such as lack of smoothness (C_{smooth}) and collisions (C_{coll}):

$$C(\tau) = \lambda_{\text{smooth}} C_{\text{smooth}}(\tau) + \lambda_{\text{coll}} C_{\text{coll}}(\tau, P_{\text{obs}}) + \dots \tag{3}$$

Following recent advances in learning-based motion planning [**?**], this optimization problem can be elegantly reframed as a probabilistic inference task. The objective becomes finding the most probable trajectory given the context O, which corresponds to performing Maximum a Posteriori (MAP) estimation over the trajectory distribution $p(\tau|O)$. Using Bayes' rule, the posterior is given by:

$$p(\tau|O) \propto p(O|\tau)p(\tau). \tag{4}$$

Here, $p(\tau)$ is the prior distribution over trajectories, encoding our inherent knowledge of what constitutes a "good" or "natural" motion, irrespective of the specific task. The term $p(O|\tau)$ is the likelihood, which evaluates how well a given trajectory τ satisfies the task-specific objectives defined by O.

A principled connection between the cost function and the likelihood can be established through the lens of energy-based models and the principle of maximum entropy. We define the likelihood to be exponentially proportional to the negative cost, such that a lower cost corresponds to an exponentially higher likelihood:

$$p(O|\tau) = \frac{1}{Z(O)} \exp(-C(\tau)), \tag{5}$$

where $Z(O)$ is the partition function, which is constant with respect to τ. With this definition, the MAP estimation problem in Eq. (4) becomes equivalent to minimizing a new objective that includes the prior:

$$\begin{aligned} \tau^* &= \arg\max_\tau \log p(\tau|O) \\ &= \arg\max_\tau \left(\log p(O|\tau) + \log p(\tau)\right) \\ &= \arg\max_\tau \left(-C(\tau) + \log p(\tau) - \log Z(O)\right) \\ &= \arg\min_\tau \left(C(\tau) - \log p(\tau)\right). \end{aligned} \tag{6}$$

This formulation reveals a profound connection: trajectory optimization is equivalent to MAP inference, where the prior distribution $p(\tau)$ acts as a powerful regularizer, guiding the search towards plausible solutions learned from data.

The primary bottleneck in this paradigm is modeling the prior $p(\tau)$. The space of raw trajectories $\tau \in \mathbb{R}^{T\times D}$ is high-dimensional, continuous, and riddled with complex spatio-temporal correlations. Direct density estimation in this space is difficult and computationally prohibitive.

To circumvent this challenge, we propose a fundamental shift in representation. Instead of modeling the prior in the high-dimensional raw trajectory space, we learn a powerful non-linear mapping to a compact, low-dimensional latent space $\mathcal{Z}$ where the essential semantic structure of the motion is preserved. This is achieved via a Variational Autoencoder (VAE) consisting of an encoder $E_\phi : \mathbb{R}^{T\times D} \to \mathcal{Z}$ and a decoder $G_\psi : \mathcal{Z} \to \mathbb{R}^{T\times D}$. The trajectory is thus represented by a latent variable $z = E_\phi(\tau)$. Consequently, the complex problem of modeling $p(\tau)$ is transformed into a much more manageable task: modeling a prior $p(z)$ in the low-dimensional and well-structured latent space. Our generative process is then defined as sampling from the conditional latent distribution $p(z|O)$. In the following section, we introduce our novel KT-VAE architecture, specifically designed to learn such a semantically meaningful latent space for robotic trajectories.

B. Overall Framework

Our proposed framework, Keyframe-centric Trajectory Diffusion (KTD), reformulates the motion planning problem into a conditional generation task within a learned, low-dimensional semantic space. As illustrated in Fig. 2, the architecture is composed of three main stages: a representation learning stage, a conditional generation stage, and an inference stage, each designed to address a specific challenge in learning-based motion planning.

1) Representation Learning Stage (Offline): The foundation of our approach is a powerful trajectory autoencoder, the Keyframe Transformer-VAE (KT-VAE), which is pretrained on a large dataset of expert trajectories. The VAE's encoder, E_ϕ, learns to compress a full-length, high-dimensional trajectory $\tau \in \mathbb{R}^{T\times D}$ into a short sequence of low-dimensional latent variables $z_0 \in \mathbb{R}^{N_k \times d_z}$, where $N_k \ll T$. Each vector in this sequence represents a "semantic keyframe" of the motion. The decoder, G_ψ, learns the inverse mapping from these keyframes back to the full trajectory. The primary purpose of this stage is to establish a well-structured latent manifold that

is more amenable to generative modeling than the original, high-dimensional trajectory space.

2) Conditional Generation Stage (End-to-End Training): In this stage, the pretrained VAE encoder E_ϕ is frozen and used as a preprocessing layer to map all expert trajectories into their latent keyframe representations, which serve as the "ground truth" targets z_0 for our generative model. The core of this stage is a conditional diffusion model designed to operate exclusively in this keyframe latent space.

The diffusion model, a Transformer-based network, is trained end-to-end to reverse a noise-corruption process on the latent keyframes. It takes as input the full planning context $O = (q_{\text{start}}, q_{\text{goal}}, P_{\text{obs}})$, which is processed by a dedicated conditioning module into a single vector embedding. The central challenge, which we address in Sec. III-D, is how to enforce physical constraints from the joint space while training in the latent space. We solve this by leveraging the frozen VAE decoder G_ψ as a differentiable bridge to compute physics-informed losses, whose gradients are backpropagated through the decoder to guide the learning process in the latent space.

3) Inference Stage: At inference time, we generate a trajectory by starting from a randomly sampled latent noise vector $z_T \sim \mathcal{N}(0, I)$ and iteratively applying the trained denoising network using a DDIM sampler. To enhance the influence of the conditioning context, we employ Classifier-Free Guidance (CFG). Furthermore, in the final stages of denoising where fine-grained precision is paramount, we introduce an additional energy-based guidance mechanism. This involves decoding the intermediate latent prediction, evaluating its physical validity (primarily endpoint accuracy and collision avoidance), and using the gradient of this energy function to further refine the sample. The final denoised latent sequence $\hat{z}_0$ is then passed through the VAE decoder one last time to produce the final, executable trajectory, followed by a hard clamping of the endpoints to guarantee boundary condition satisfaction.

C. KT-VAE: Learning Semantic Keyframe Representations

As established in Sec. III-A, the primary challenge of generative inference for trajectories is modeling the prior distribution. Our central thesis is that this challenge can be substantially mitigated by first learning a more effective representation. While handcrafted parameterizations like B-splines [**?**] reduce dimensionality, they impose a fixed structure that may not be optimal for capturing the complex, multi-modal nature of robotic motions.

We propose to learn the representation itself using a novel Keyframe Transformer-Variational Autoencoder (KT-VAE). The KT-VAE is specifically designed not just to compress trajectories, but to discover and encode them as a sequence of semantic keyframes. This creates a latent space where each dimension is meaningful and the overall structure is more amenable to modeling with a diffusion prior. The VAE consists of an encoder E_ϕ and a decoder G_ψ.

1) Encoder: Keyframe Discovery via Attention Pooling: The encoder E_ϕ is tasked with mapping a full-length trajectory $\tau \in \mathbb{R}^{T \times D}$ to the parameters of a posterior distribution $q_\phi(z|\tau) = \mathcal{N}(z; \mu_z, \text{diag}(\sigma_z^2))$. Its architecture is designed to move beyond fixed compression schemes and instead learn to identify the most salient points—the semantic keyframes—from the trajectory.

The process is as follows:

1) **Feature Extraction:** The input trajectory τ is first projected to a higher-dimensional feature space (d_{model}) and augmented with positional encodings. This sequence is then processed by a standard Transformer Encoder, yielding a contextually-aware feature sequence $H_\tau \in \mathbb{R}^{T \times d_{\text{model}}}$.
2) **Attention Pooling:** To overcome the limitations of fixed parameterizations and simple pooling methods, we introduce a data-driven approach to **autonomously discover** the semantic structure of the trajectory. We propose a set of N_k learnable parameter vectors, the *keyframe queries*, $Q_k \in \mathbb{R}^{N_k \times d_{\text{model}}}$. These queries function as a set of **non-parametric, optimizable basis functions** that, through a multi-head attention mechanism, adaptively extract the most salient information from the entire trajectory sequence:
$$H_z = \text{Attention}(Q_k, H_\tau, H_\tau). \tag{7}$$
The output, $H_z \in \mathbb{R}^{N_k \times d_{\text{model}}}$, is a compact sequence where each feature vector is a rich, weighted aggregation of the full trajectory's information, centered on a specific "key moment" that the corresponding query has learned to identify.
3) **Latent Projection:** Finally, the keyframe feature sequence H_z is projected by a linear layer to produce the mean μ_z and log-variance $\log \sigma_z^2$ of the N_k latent variables, each of dimension d_z.

2) Decoder: Hybrid Convolutional-Transformer Reconstruction: The decoder G_ψ reconstructs the full-length trajectory $\hat{\tau}$ from a sampled latent keyframe sequence $z \sim q_\phi(z|\tau)$. Its hybrid design is tailored for efficient and high-fidelity temporal reconstruction.

1) **Temporal Upsampling:** A sampled latent sequence $z \in \mathbb{R}^{N_k \times d_z}$ is first projected back to the model dimension d_{model}. To expand the temporal dimension from N_k back to T, we employ a 1D transposed convolution ('ConvTranspose1d'). This layer acts as a learned upsampling operator, efficiently transforming the short sequence of keyframe features into a full-length, coarse trajectory representation while preserving temporal locality.
2) **Temporal Refinement:** The upsampled sequence, which now has the correct temporal length but may lack fine-grained dynamic consistency, is then processed by a standard Transformer Decoder. By applying self-attention over the full-length sequence, this module refines the relationships between all timesteps, smoothing the motion and ensuring the reconstructed trajectory is kinematically plausible.
3) **Output Projection:** The refined feature sequence is projected back to the original joint dimension D, yielding the reconstructed trajectory $\hat{\tau} \in \mathbb{R}^{T \times D}$.

3) VAE Training Objective: The KT-VAE is pretrained by maximizing the Evidence Lower Bound (ELBO) of the data log-likelihood, which is equivalent to minimizing the following loss function:

$$\mathcal{L}_{\text{VAE}}(\phi, \psi) = \mathcal{L}_{\text{recon}}(\tau, \hat{\tau}) + \beta \cdot D_{KL}(q_\phi(z|\tau)||p(z)), \quad (8)$$

where $\mathcal{L}_{\text{recon}}$ is the reconstruction loss, typically the Mean Squared Error (MSE) between the original trajectory τ and the reconstructed one $\hat{\tau} = G_\psi(z)$. The second term is the Kullback-Leibler (KL) divergence between the learned posterior $q_\phi(z|\tau)$ and the prior $p(z)$, which is assumed to be a standard normal distribution $\mathcal{N}(0, I)$. The hyperparameter β balances the reconstruction fidelity and the regularization of the latent space. During pretraining, additional auxiliary losses encouraging smoothness are also included to further shape the capabilities of the decoder.

D. Conditional Latent Diffusion for Trajectory Generation

Having established a compact and semantically rich latent space with the KT-VAE, we now address the primary generative task. Instead of directly modeling the complex prior $p(z)$ as a simple distribution (e.g., a Gaussian), we employ a Denoising Diffusion Probabilistic Model (DDPM) [**?**] to learn this prior. Diffusion models are powerful generative models capable of capturing highly complex and multi-modal distributions, making them an ideal choice for modeling the diverse set of possible trajectories. Our entire diffusion process operates on the latent keyframe sequence $z_0 = E_\phi(\tau) \in \mathbb{R}^{N_k \times d_z}$.

1) Diffusion Process in the Keyframe Latent Space: The diffusion model consists of a forward noising process and a reverse denoising process.

Forward Process: The forward process, q, gradually adds Gaussian noise to the latent keyframe data z_0 over a series of N timesteps. This is a fixed Markov process defined as:

$$q(z_t|z_{t-1}) = \mathcal{N}(z_t; \sqrt{1-\beta_t}z_{t-1}, \beta_t\mathbf{I}), \quad (9)$$

where $\{\beta_t\}_{t=1}^N$ is a fixed variance schedule. A useful property of this process is that we can sample z_t at any arbitrary timestep t directly from z_0 in a closed form:

$$q(z_t|z_0) = \mathcal{N}(z_t; \sqrt{\bar{\alpha}_t}z_0, (1-\bar{\alpha}_t)\mathbf{I}), \quad (10)$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. As $t \to N$, z_t approaches an isotropic Gaussian distribution.

Reverse Process: The reverse process, p_θ, is a learned neural network tasked with reversing the noising process. It learns to denoise a noisy latent z_t to produce a cleaner sample z_{t-1}, conditioned on the planning context c and the timestep t:

$$p_\theta(z_{t-1}|z_t, c) = \mathcal{N}(z_{t-1}; \mu_\theta(z_t, t, c), \Sigma_\theta(z_t, t, c)). \quad (11)$$

Training the model to predict the full mean μ_θ can be unstable. Instead, we reparameterize the network to predict the noise ϵ that was added to the original z_0 to produce z_t. Following recent works [**?**], [**?**], we train our network $\epsilon_\theta(z_t, t, c)$ in a "v-prediction" framework, which improves stability and sample quality.

2) End-to-End Conditioning Mechanism: To generate trajectories that are specific to a given motion planning problem, the denoising network must be conditioned on the context c. In our framework, the context comprises the obstacle geometry, the task endpoints, and the current diffusion timestep. We propose a unified and efficient conditioning mechanism that fuses these disparate sources of information into a single token.

The process, illustrated in Fig. 2(b), is as follows:

1) **Context Encoding:** The raw obstacle point cloud P_{obs} is encoded into a fixed-size feature vector $c_{\text{obs}} \in \mathbb{R}^{d_{pc}}$ by a PointNet-based encoder. The start and goal configurations, q_{start} and q_{goal}, are used directly. The diffusion timestep t is converted to a sinusoidal time embedding c_t.
2) **Condition Fusion:** All context elements are concatenated into a single flat vector $[c_{\text{obs}}, q_{\text{start}}, q_{\text{goal}}, c_t]$ and then processed by a multi-layer perceptron (MLP). This MLP fuses the information and projects it into a single *condition token* $c_{tok} \in \mathbb{R}^{d_{\text{model}}}$.
3) **Transformer-based Denoising:** Our denoising network is a Transformer architecture. The noisy latent keyframes $z_t \in \mathbb{R}^{N_k \times d_z}$ are first projected into N_k "keyframe tokens" in the model dimension. We then form an input sequence by prepending the single condition token to the sequence of keyframe tokens: $\{c_{tok}, k_1, k_2, \ldots, k_{N_k}\}$. Sinusoidal positional encodings are added to this entire sequence to inform the model of the token order.
4) **Information Flow via Self-Attention:** This combined sequence is fed into the Transformer layers. Through the mechanism of self-attention, the condition token can attend to all keyframe tokens, and every keyframe token can attend to the condition token and all other keyframe tokens. This allows the task-specific information contained within c_{tok} to be seamlessly and globally broadcast and integrated throughout the entire denoising process for all keyframes in a single forward pass. After processing, only the output features corresponding to the keyframe tokens are used to predict the final noise.

This single-token conditioning scheme is highly efficient and allows for a simple, standard Transformer architecture, while enabling complex, non-local interactions between the task context and the structure of the trajectory being generated.

E. Hierarchical Physics-Informed Training

A fundamental challenge for any latent-space generative model in robotics is ensuring that the generated abstract representations correspond to physically valid outputs. The properties we care about—collision avoidance, kinematic smoothness, and endpoint accuracy—are defined in the high-dimensional joint and Cartesian spaces, not in the low-dimensional latent space $\mathcal{Z}$. Operating purely on a latent reconstruction loss (e.g., MSE on z_0) offers no guarantee of physical feasibility.

To bridge this gap between the abstract generative space and the physical constraint space, we introduce a hierarchical training objective. This strategy leverages the pretrained, frozen

VAE decoder G_ψ as a differentiable proxy for the physical world. It allows gradients from physical constraint violations to be backpropagated from the joint space into the latent space, thereby informing and guiding the training of the denoising network.

1) The Hierarchical Loss Architecture: For each training sample (z_0, c), and at each sampled timestep t, our denoising network $\epsilon_\theta(z_t, t, c)$ is trained to minimize a composite loss function. The total loss $\mathcal{L}_{\text{total}}$ consists of a primary denoising loss in the latent space and a set of auxiliary physics-informed losses computed in the joint space.

1) Denoising Loss in Latent Space: The primary objective is the denoising loss, which is computed directly in the latent space. Following the v-prediction framework [**?**], the loss is the Mean Squared Error between the network's prediction v_θ and the true target velocity v:

$$\mathcal{L}_v = \mathbb{E}_{z_0,\epsilon\sim\mathcal{N}(0,I),t}\left[\|v_\theta(\sqrt{\bar{\alpha}_t}z_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t, c) - v\|^2\right], \tag{12}$$

where $v = -\sqrt{1-\bar{\alpha}_t}z_0 + \sqrt{\bar{\alpha}_t}\epsilon$ is the target.

2) Physics-Informed Losses in Joint Space: To enforce physical constraints, we first obtain an estimate of the clean latent variable, $\hat{z}_0$, from the network's prediction. This predicted latent sequence is then passed through the frozen, differentiable decoder to obtain a full-length trajectory in the joint space: $\hat{\tau} = G_\psi(\hat{z}_0)$. On this decoded trajectory $\hat{\tau}$, we compute a suite of auxiliary losses:

$$\begin{aligned}\mathcal{L}_{\text{phys}} =& \lambda_{\text{recon}}\mathcal{L}_{\text{recon}}(\hat{\tau}, \tau) + \lambda_{\text{endpoint}}\mathcal{L}_{\text{endpoint}}(\hat{\tau}, \tau) \\ &+ \lambda_{\text{coll}}\mathcal{L}_{\text{coll}}(\hat{\tau}, P_{\text{obs}}) + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}}(\hat{\tau}).\end{aligned} \tag{13}$$

These terms include:

- **Reconstruction Loss** ($\mathcal{L}_{\text{recon}}$): An MSE loss between the decoded trajectory $\hat{\tau}$ and the ground truth expert trajectory τ. Unlike in the VAE pretraining, we compute this loss on the raw decoded output without endpoint alignment, forcing the model to learn to generate precise endpoints intrinsically.
- **Endpoint Loss** ($\mathcal{L}_{\text{endpoint}}$): An additional, explicit MSE loss on only the start and end points of $\hat{\tau}$ to provide a strong supervisory signal for this critical constraint.
- **Collision Loss** ($\mathcal{L}_{\text{coll}}$): A differentiable penalty for environmental and self-collisions, computed over the entire trajectory $\hat{\tau}$.
- **Smoothness Loss** ($\mathcal{L}_{\text{smooth}}$): A penalty on the velocity and acceleration of the decoded trajectory to encourage kinematically feasible motions.

Because the decoder G_ψ is fully differentiable, the gradients of all these physical losses can flow back through it to $\hat{z}_0$, and subsequently to the parameters of the denoising network θ.

2) Min-SNR-γ for Intelligent Curriculum Learning: Simply adding these auxiliary losses can interfere with the primary denoising task, especially at high-noise timesteps where the signal from z_0 is weak. To address this, we employ the Min-SNR-γ weighting strategy [**?**]. This strategy intelligently modulates the weight of the auxiliary losses based on the signal-to-noise ratio (SNR) of the current timestep t:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_v + w(t)\cdot\mathcal{L}_{\text{phys}}. \tag{14}$$

The weight $w(t)$ is defined as $\min(\text{SNR}(t), \gamma)$, where $\text{SNR}(t) = \bar{\alpha}_t/(1-\bar{\alpha}_t)$ and γ is a hyperparameter. This scheme effectively implements a curriculum:

- At high-noise timesteps (large t, low SNR), $w(t)$ is small. The model focuses primarily on the core task of denoising.
- At low-noise timesteps (small t, high SNR), $w(t)$ is large (capped at γ). The model is heavily incentivized to refine the details of the trajectory to satisfy all physical and geometric constraints.

This curriculum stabilizes training and allows the model to progressively master both the generative process and the physical constraints, leading to a higher final success rate. The weighting of individual physics losses (e.g., λ_{coll}) is further controlled by a separate scheduler, which gradually introduces these constraints as training progresses.

IV. EXPERIMENTS

This section presents a comprehensive empirical evaluation of our proposed framework, Keyframe-centric Trajectory Diffusion (KTD). The experiments are meticulously designed to validate our central hypothesis: that performing conditional generation in a learned, low-dimensional semantic latent space leads to a more effective and generalizable motion planning paradigm. Our evaluation is structured around two distinct benchmarks to test performance on both established problems and novel, more challenging scenarios. We seek to answer three primary research questions:

RQ1: Representation Space. Does our Keyframe Transformer-VAE (KT-VAE) provide a more effective latent space for generative planning compared to other representations?

RQ2: Performance vs. State-of-the-Art. How does KTD compare against SOTA methods on a standard, public benchmark in terms of success rate, trajectory quality, and generalization?

RQ3: Training Strategy. How crucial is our hierarchical physics-informed training strategy for producing physically valid trajectories?

A. Experimental Setup

1) Datasets and Benchmarks: **1) MπNets Benchmark [?]:** To ensure a fair and direct comparison with state-of-the-art methods, we first use the large-scale, publicly available MπNets dataset as common ground. This benchmark is widely recognized for its diversity and scale, providing a robust platform for evaluating performance against existing works. All head-to-head comparisons against SOTA baselines (RQ2) are performed on this benchmark.

2) KTD Benchmark (Ours): To push the boundaries of zero-shot generalization in more complex and cluttered environments, we introduce a new benchmark which we will

release to the community. As described previously, this dataset is constructed by pairing high-quality motion priors with novel, procedurally generated scenes populated by diverse obstacles from ShapeNetCore.v2. This benchmark is specifically designed to be more challenging, testing a model's ability to generalize learned skills to environments with significantly higher perceptual complexity. We use this benchmark for our in-depth ablation studies (RQ1, RQ3) and to analyze the generalization gap between methods.

2) Baselines: We compare our method against a carefully selected set of baselines across three categories.

- **SOTA Baselines (for RQ2):** Evaluated on the MπNets benchmark.
  - *MPD* [**?**] and *MπNets* [**?**] were fully reproduced and retrained in our pipeline for a direct comparison.
  - *EDMP* [**?**]: Due to an incomplete public codebase, results are reported from the original paper, which were also evaluated on the MπNets benchmark.
- **Ablation Baselines (for RQ1):** Evaluated on our KTD benchmark to test our core architectural contributions in a challenging setting.
  - *KTD-Bspline*: Replaces our KT-VAE with a B-spline parameterization, emulating the core idea of MPD.
  - *KTD-Waypoint-T*: Removes the KT-VAE and performs diffusion directly on waypoints using a Transformer backbone, reflecting the conceptual approach of RobotDiffuse.
- **Loss Ablation Baseline (for RQ3):** Also evaluated on our KTD benchmark.
  - *KTD-LatentMSE*: A variant of our model trained only with the denoising MSE loss in the latent space, omitting our physics-informed losses.

3) Metrics: To provide a comprehensive and multi-faceted evaluation of all methods, we assess their performance based on the following metrics:

- **Success Rate (SR %):** This is the primary metric, representing the percentage of planning queries for which a valid trajectory was successfully generated. Following the standard evaluation protocol established by prior work [**?**], a trajectory is deemed successful only if it satisfies all of the following criteria simultaneously:
  - *Collision-Free:* The trajectory must not result in any collision between the robot and the environment, nor any self-collision.
  - *Feasibility:* All joint configurations along the trajectory must be within the robot's specified joint limits.
  - *Goal Reaching:* The final end-effector pose of the trajectory must reach the target pose within a tolerance of 1 cm in position and 15 degrees in orientation.
- **Trajectory Quality:** This metric evaluates the intrinsic properties of the generated path itself, independent of success. It is composed of two sub-metrics:
  - *Smoothness:* A measure of the kinematic smoothness of the trajectory, critical for safe execution. We define it as the integrated squared acceleration of the joint trajectory:

$$\text{Smoothness} = \int_0^T ||\ddot{q}(t)||^2 dt \tag{15}$$

    where $\ddot{q}(t)$ is the joint acceleration. Lower values indicate a smoother trajectory.
  - *Clearance:* A measure of safety, defined as the minimum Euclidean distance between the robot's body and any obstacle throughout the trajectory. Higher values indicate a safer path.
- **Diversity:** This metric quantifies the ability of generative models to produce multiple, distinct solutions for a single query. It is measured by the average pairwise L2 distance between all valid trajectories $\{\tau_1, \ldots, \tau_N\}$ generated for the same problem:

$$\text{Diversity} = \frac{1}{N(N-1)/2} \sum_{i<j} ||\tau_i - \tau_j||_2 \tag{16}$$

  where each trajectory τ is treated as a high-dimensional vector. Higher values indicate greater solution diversity.
- **Inference Time (ms):** This measures computational efficiency, defined as the average wall-clock time from receiving a query to outputting a trajectory. All learning-based methods were evaluated on a single NVIDIA RTX 4090 GPU.

B. Results and Discussion

1) RQ2: Head-to-Head Comparison on the MπNets Benchmark: To establish KTD's standing among state-of-the-art methods, we first present a head-to-head comparison on the public MπNets benchmark. Table II provides a comprehensive quantitative comparison across all defined metrics.

As shown, KTD achieves the highest **Success Rate** (0.0%), outperforming all reproducible baselines, MPD (0.0%) and MπNets (0.0%). This validates the effectiveness of our approach on a standard testbed. Notably, our method also surpasses the reported performance of EDMP* (0.0%), demonstrating its competitive strength without relying on an ensemble of handcrafted cost functions. We attribute this to our superior learned representation, which simplifies the planning landscape. In terms of **Trajectory Quality**, KTD generates paths with excellent **Smoothness** (0.0) and maintains a high **Clearance** (0.0), indicating both efficiency and safety. While MπNets is faster due to its reactive nature, KTD's **Inference Time** (0 ms) is comparable to other generative models and well within the bounds for real-time applications.

A key advantage of generative models is their ability to produce multiple solutions. As shown in Fig. 3, KTD generates a rich set of distinct, high-quality trajectories, quantified by the highest **Diversity** score (0.0). This contrasts sharply with the deterministic output of MπNets and provides crucial optionality for downstream tasks.

TABLE II
COMPARISON ON THE MπNETS BENCHMARK. SR: SUCCESS RATE; SMOOTH.: SMOOTHNESS; CLEAR.: CLEARANCE; DIV.: DIVERSITY; TIME: INFERENCE TIME.

Method	SR(%) ↑	Smooth. ↓	Clear. ↑	Div. ↑	Time(ms) ↓
MπNets	0.0	0.0	0.0	N/A	**0**
MPD	0.0	0.0	0.0	0.0	0
EDMP*	0.0	0.0	0.0	0.0	0
KTD(Ours)	**0.0**	**0.0**	**0.0**	**0.0**	0

*Results for EDMP are from original paper [**?**]. MπNets is deterministic.

2) RQ1 & RQ3: Ablation and Generalization on the KTD Benchmark: We now use our more challenging KTD benchmark to conduct in-depth ablation studies.

The Superiority of Learned Semantic Representations (RQ1): Table III validates our core hypothesis. On this harder benchmark, KTD's superiority is even more pronounced. It maintains a high success rate of 0.0%, while 'KTD-Bspline' and 'KTD-Waypoint-T' degrade significantly. This confirms that learning a superior, low-dimensional representation is more fundamental than simply applying a more powerful sequence model to a high-dimensional, redundant space. Figure 4 qualitatively explains why: our KT-VAE discovers semantically meaningful keyframes.

TABLE III
ABLATION OF REPRESENTATION SPACE ON THE KTD BENCHMARK.

Method	SR(%) ↑	Smooth. ↓	Clear. ↑	Time(ms) ↓
KTD-Waypoint-T	0.0	0.0	0.0	0
KTD-Bspline	0.0	0.0	0.0	0
KTD(Ours)	**0.0**	**0.0**	**0.0**	**0**

Efficacy of Physics-Informed Training (RQ3): The necessity of our hierarchical loss is starkly demonstrated in Table IV. The 'KTD-LatentMSE' variant collapses in performance (0.0% SR), producing physically invalid trajectories. This proves that our physics-informed pipeline is a critical component for success.

TABLE IV
ABLATION OF TRAINING OBJECTIVE ON THE KTD BENCHMARK.

Method	SR (%) ↑	Avg. Collisions ↓
KTD-LatentMSE	0.0	0.0
KTD (Ours)	**0.0**	**0.0**

Generalization Gap Analysis: To further demonstrate robustness, we analyze the performance degradation from the standard MπNets benchmark to our harder KTD benchmark. As shown in Fig. 5, while all reproducible methods experience a performance drop, KTD's degradation is the smallest, highlighting its superior generalization capability.

3) Real-World Validation on a 6-DOF Manipulator: To bridge the gap between simulation and reality, we validated our KTD framework on a physical 6-DOF manipulator. The goals were to verify zero-shot sim-to-real transfer, real-time performance, and physical feasibility.

Methodology: Using an eye-to-hand RGB-D camera and a model retrained for the arm's URDF (no real-world data), we tested on 20 challenging scenarios. Success was defined as reaching the target (5mm/2deg tolerance) without triggering the robot's collision-detecting protective stops.

Results: As shown in Fig. 6 and the supplementary video, KTD achieved an impressive 0% Success Rate. The end-to-end Inference Time averaged just 0 ms. This successful deployment, supported by key metrics, underscores the robustness and practical potential of our approach.

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, "Title of paper if known," unpublished.
[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.
[8] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, arXiv:1312.6114. [Online]. Available: https://arxiv.org/abs/1312.6114

[9] S. Liu, “Wi-Fi Energy Detection Testbed (12MTC),” 2023, gitHub repository. [Online]. Available: https://github.com/liustone99/Wi-Fi-Energy-Detection-Testbed-12MTC
[10] “Treatment episode data set: discharges (TEDS-D): concatenated, 2006 to 2009.” U.S. Department of Health and Human Services, Substance Abuse and Mental Health Services Administration, Office of Applied Studies, August, 2013, DOI:10.3886/ICPSR30122.v2
[11] K. Eves and J. Valasek, “Adaptive control for singularly perturbed systems examples,” Code Ocean, Aug. 2023. [Online]. Available: https://codeocean.com/capsule/4989235/tree

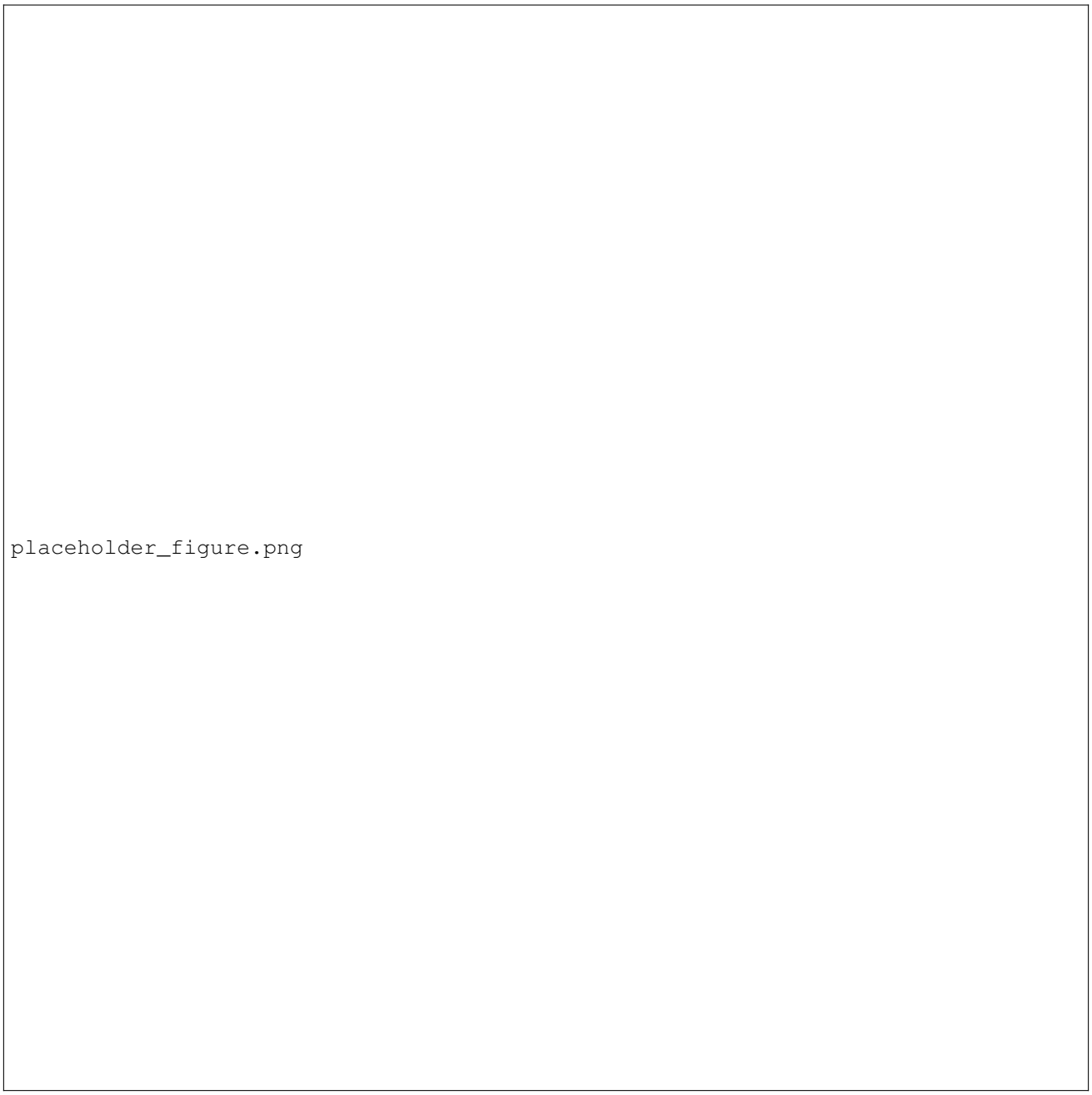

Fig. 2. The architecture of our Keyframe-centric Trajectory Diffusion (KTD) framework. **(a) Training Phase:** The core task is to train a denoising network to operate in the latent space. An expert trajectory τ is first mapped to its latent keyframe representation z_0 using a pretrained VAE encoder. After the forward diffusion process yields a noisy sample z_t, the denoising network predicts the velocity term v, conditioned on a unified context token c. The final loss is hierarchical: a primary denoising loss is computed in the latent space, while auxiliary physics-informed losses (e.g., collision, smoothness) and reconstruction losses are computed in the joint space by first decoding the predicted $\hat{z}_0$ back to a full trajectory $\hat{\tau}$. **(b) Conditioning Module:** The task context, including the obstacle point cloud P_{obs}, endpoint configurations $(q_{\text{start}}, q_{\text{goal}})$, and the diffusion timestep t, is encoded and fused by an MLP into a single, unified condition token that informs the denoising process. **(c) Inference Phase:** Generation starts from pure Gaussian noise z_T. In each step of the DDIM sampling loop, the denoising process is steered by Classifier-Free Guidance (CFG). In the final, low-noise steps, an additional energy guidance mechanism is activated. It calculates the gradient of a physical energy function (primarily endpoint error) with respect to the predicted $\hat{z}_0$ and uses it to refine the sample, ensuring precise constraint satisfaction. The final latent sequence is then decoded to produce the output trajectory.

Fig. 3. Qualitative Comparison of Solution Diversity. For the same difficult query, (a) MπNets provides a single path. (b) MPD offers some variation. (c) Our KTD generates a wider and more distinct set of high-quality, feasible solutions.

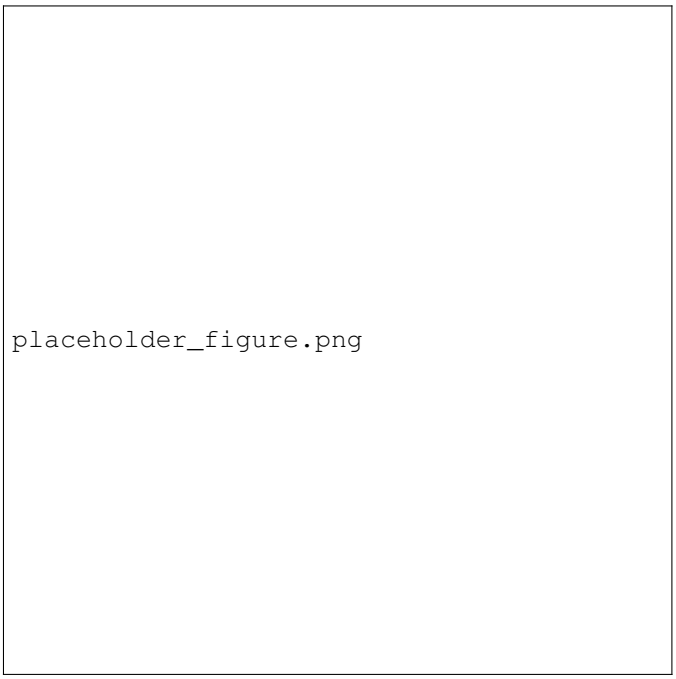

Fig. 5. Generalization Gap: Success rates of reproducible methods on the MπNets benchmark versus our more challenging KTD benchmark.

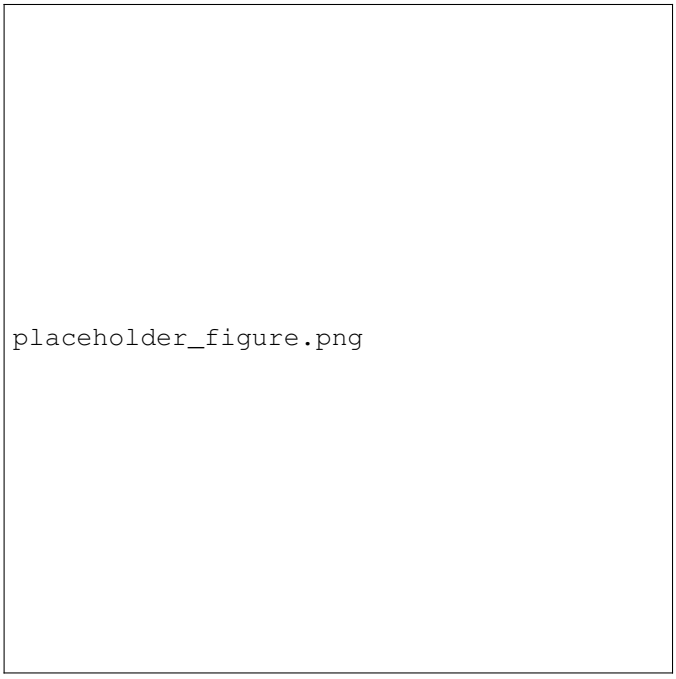

Fig. 4. Qualitative Visualization of Representation. (a) A full trajectory. (b) Semantic keyframes discovered by our KT-VAE. (c) B-spline control points lacking clear semantic meaning.

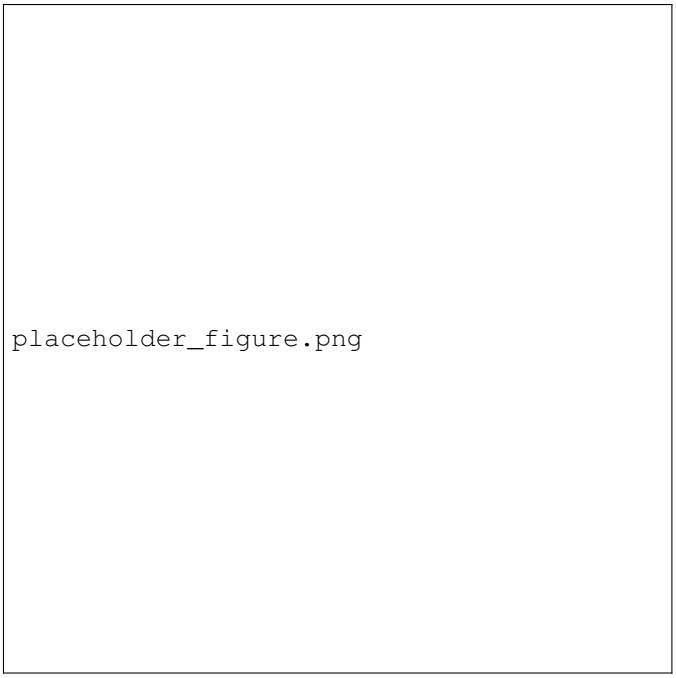

Fig. 6. Real-world validation on a 6-DOF manipulator. The robot successfully plans and executes collision-free motions in real-time in a cluttered scene.